

Fine-Grained Tracing Without Binary Instrumentation

or: Don't Be a Pinhead, Think Outside the Bochs

Ryan MacArthur
iSIGHT Partners Lab
April 28th 2011

About Me

- Don't believe anything I say
- Graduated in 2008
- Symantec
 - Security Response team
- Interviewed with iSIGHT at BH Vegas 2009
- Started *work* in October 2009



shellcode

Execution Traces

- Extremely valuable
 - Code Coverage
 - What code wasn't hit?
- Why
 - Exploitability (why did code crash)
 - Obfuscated Malware analysis (what did code do)
 - Reverse engineering (how did code execute)

Tracing Types

- :
- Coarse Grained Tracing
 - Syscalls
 - Call graphs!
- Fine Grained Tracing
 - Instructions
 - Basic Blocks!

Static/Dynamic Instrumentation

- PIN
- DynamoRIO
- Valgrind
- .NET profiling API
- ...



Tracing Options

- APIs
 - ptrace, DebugActiveProcess, WaitForDebugEvent
- Debuggers!
 - EDB, WinDbg, OllyDBG
- Specialized tools
 - Process Stalker (paimei)
 - COSEINC TRACE SUITE

Common Methods

- Trap EFLAGS
- Inject Breakpoints
- Hardware Breakpoints
- Hook Functions
- ?

WTF

- Why Am I up here
 - Problem solved, right?

Limits

- Checksumming Code
 - Self Modifying Code
 - Debugger Detection
 - Virtualization Detection
- Companies likely pay analysts to handle these cases by hand. Fuck that.
- I'm impatient, and potentially we have millions of samples to trace, or to trace a few samples` millions of times :o

What's in your sandbox

401061 LoadLibraryA(User32)

4011b8 mdll kernel32> cmp byte [eax],0xe8 7c801d77 LoadLibraryA READ

4011bd mdll kernel32> cmp byte [eax],0xe9 7c801d77 LoadLibraryA READ

4011c2 mdll kernel32> cmp byte [eax],0xeb 7c801d77 LoadLibraryA READ

Lookie Lookie, I've got Hookie.

```
seg000:000005ED Call_Api_Hook_Check proc near          ; CODE XREF: sub_A8+38
seg000:000005ED                                     ; sub_A8+206 p ...
seg000:000005ED     cmp     byte ptr [eax], 0E8h ; 'F'
seg000:000005F0     jz      short has_hook
seg000:000005F2     cmp     byte ptr [eax], 0E9h ; 'T'
seg000:000005F5     jz      short has_hook
seg000:000005F7     cmp     byte ptr [eax], 0EBh ; 'd'
seg000:000005FA     jnz     short to_jump
seg000:000005FC
seg000:000005FC     has_hook:                               ; CODE XREF: Call_Api_
seg000:000005FC                                     ; Call_Api_Hook_Check+
seg000:000005FC     cmp     dword ptr [eax+5], 90909090h
seg000:00000603     jz      short to_jump
seg000:00000605     mov     edi, edi
seg000:00000607     push    ebp
seg000:00000608     mov     ebp, esp
seg000:0000060A     lea     eax, [eax+5]
```

Party at Ring 0

- Exclusive?
 - hipster

Intel to the rescue

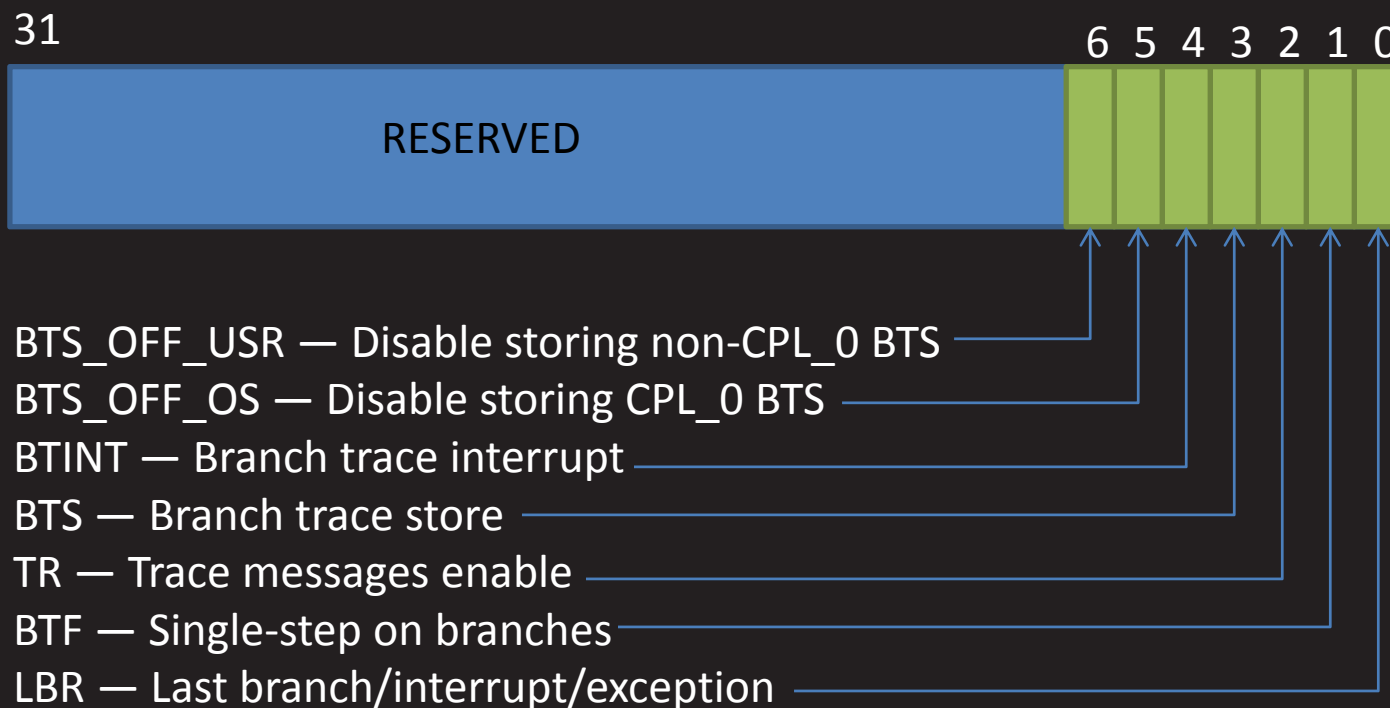
- Last Branch Recording (LBR)
- Lots of fun Debugging/Performance stuff to play with (Volume 3B)

LBR

- LBR Stack
- Trap On Branch
- Branch Trace Store
 - Cyclical buffer or interrupt
 - Local APIC
 - Trace only userland or trace OS

Last branch recording facilities (netburst)

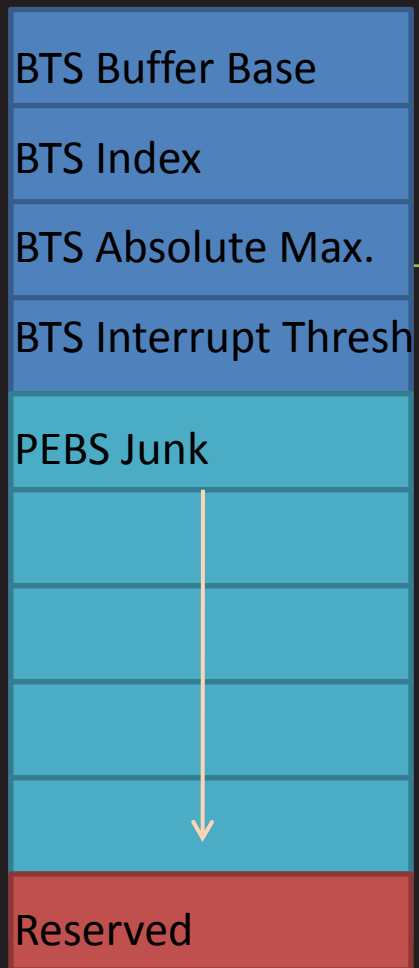
- MSR_DEBUGCTLA



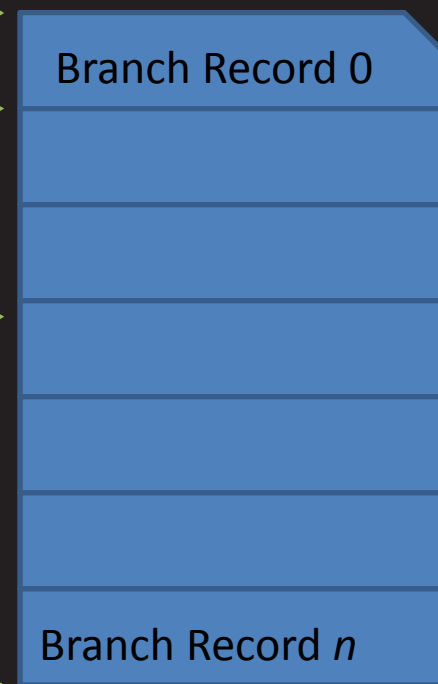
Debug Store

IA32_DS_AREA MSR

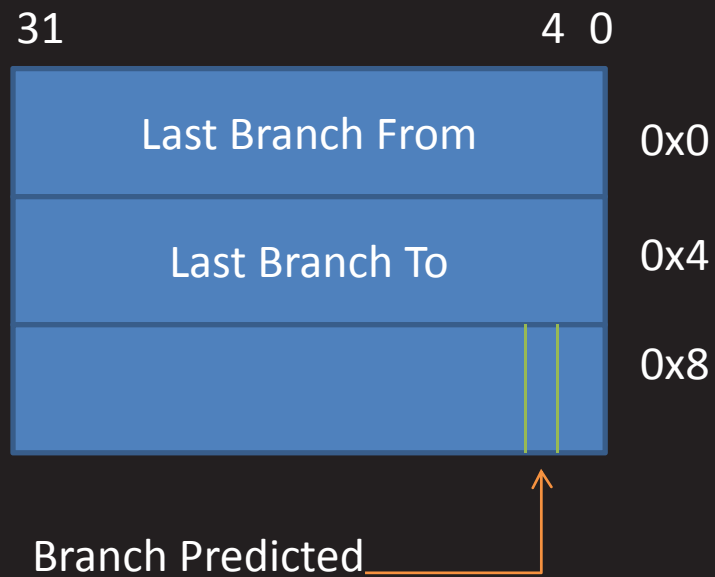
DS Buffer Management Area



BTS BUFFER



Branch Record



Thanks msft

```
.text:00405933      _KiDispatchInterrupt00 endp
.text:00405933
.text:00405934      ; -----
.text:00405934  8D A4 24 00 00 00 00      lea     esp, [esp+0]      ; Load Effective Address
.text:00405938  8D A4 24 00 00 00 00      lea     esp, [esp+0]      ; Load Effective Address
.text:00405942
.text:00405942      ; ===== S U B R O U T I N E =====
.text:00405942
.text:00405942      SwapContext      proc near      ; CODE XREF: KiUnlockDispatcherDatabase(x)+99↑p
.text:00405942                                     ; KiSwapContext(x)+2A↑p ...
.text:00405942
.text:00405942      ; FUNCTION CHUNK AT .text:00405AC9 SIZE 00000034 BYTES
.text:00405942
.text:00405942  000 0A C9                or      cl, cl            ; Logical Inclusive OR
.text:00405944  000 26 C6 46 2D 02      nov     byte ptr es:[esi+2Dh], 2
```

Jump gate

```
804dce62 c3          ret
804dce63 90          nop
804dce64 8da4240000ac9 lea      esp,[esp-36F60000h]
804dce6b eb0c      jmp      nt!SwapContext+0x2 (804dce79)
804dce6d 90          nop
804dce6e e9fdf79539    jmp      miser+0x670 (b9e3c670)
804dce73 90          nop
804dce74 90          nop
804dce75 90          nop
804dce76 90          nop
nt!SwapContext:|
804dce77 ebf5      jmp      nt!KiDispatchInterrupt+0xdc (804dce6e)
804dce79 26c6462d02    mov     byte ptr es:[esi+2Dh],2
```

My Windows Implementation

- Mimics malware
 - Userland exe drops kernel driver, patches it with address of swapcontext() and installs it
 - Driver hooks swapcontext() and the fun begins
 - Walk kernel objects and turn on tracing for threads of interest
 - KTHREAD -> ETHREAD

Userland Component

- Registers processes of interest
- Consumes trace data
 - Event Driven

USER

INTERESTING
THING

obsessor.exe

EXECUTIVE

THREAD QUEUE



SwapContext()



miser.sys



CPU n

MSR

APIC n

LVT

Caveats

- There must be separate save areas, buffers, and state for each processor in an MP system.
 - Just do these per thread, makes post-processing easier
- On my implementation, ~1000 records are written between ISR -> DPC to copy them to userland

Caveats

- MSR's are ... well, model specific
- ISR is supposed to (amongst other things):
 - The ISR must clear the mask bit in the performance counter LVT entry
 - Check
 - The Pentium 4 Processor and Intel Xeon Processor mask PMIs upon receiving an interrupt. Clear this condition before leaving the interrupt handler.
 - /me shrugs

Caveats

- Don't lose trace records
 - Double buffering rocks
 - (thanks evan)

Performance

- I should have used PEBS to measure, but ran out of time
- Used jump heavy code as sample
- Timing resolution is millisecond :/
- Miser, Coseinc Trace Suite, Pin inscount2
 - Could not get saffron from author :(

perf_test.c

```
#include <windows.h>

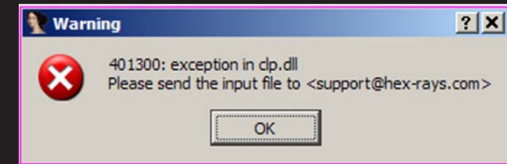
void main(int argc, char **argv)
{
    unsigned int ticks,i,j;

    ticks = GetTickCount();

    for(i=0 ; i< atoi(argv[1]); i++)
    {
        if (i == rand()) continue;
        if (i == rand()) continue;
        ...
        if (i == rand()) continue; ← 100,000th
        j = i ^ ticks;
    }
    printf("ticks %u\n", (GetTickCount() - ticks),j);
}
```

JNZzzzzz!

- IDA temporarily chokes
- Crashed HexRays Decompiler
- Should be enough torture for my tracer



Perf. Results

- $\bar{x} =$ 10 runs of perf_test, argv[1] = 1
- Native: $\bar{x} = 0ms$
- Miser: $\bar{x} = 0ms$
- Coseinc: $\bar{x} = 2,391ms$
- Inscout1: $\bar{x} = 38,406ms$
- Process Stalker (BTF) $\bar{x} = 43,672ms$
- Single Step $\bar{x} = 84,656ms$

Perf. Results

- $\bar{x}=10$ runs of perf_test, argv[1] = 10,000
- Native: $\bar{x}=15,724ms$
- Miser: $\bar{x}=15,811ms$
- Coseinc: $\bar{x}=17,359ms$
- Inscout1: $\bar{x}=75,250ms$

AddressOfEntryPoint 000012a0

MISER (BTS -- BTINT)

from	to	predicted
0x7c9011a4,	0x00401360,	0x00000000
0x804de865,	0x00401360,	0x00000010
0x00401385,	0x00401487,	0x00000000
0x0040149d,	0x00401c10,	0x00000010
0x00401c1c,	0x00401c56,	0x00000000
0x00401c5d,	0x00401c71,	0x00000000
0x00401c78,	0x00401d98,	0x00000010
0x00401d98,	0x7c809fa1,	0x00000000
0x7c809fb8,	0x00401c7d,	0x00000010
0x00401c80,	0x00401c5f,	0x00000010
0x00401c70,	0x004014a2,	0x00000010
0x004014a2,	0x0040138b,	0x00000010
0x00401394,	0x7c9011a7,	0x00000000
0x7c9011a4,	0x004014d0,	0x00000000
0x004014f0,	0x7c9011a7,	0x00000010
0x7c816d4c,	0x004012a0,	0x00000000
0x004012ad,	0x77c3537c,	0x00000000
0x77c3538a,	0x004012b3,	0x00000010
0x004012b3,	0x00401150,	0x00000010
0x804de865,	0x00401157,	0x00000010
0x0040115e,	0x00401260,	0x00000000
0x00401276,	0x00401360,	0x00000000
0x00401382,	0x00401397,	0x00000000
0x004013a6,	0x0040138b,	0x00000000
0x00401394,	0x00401278,	0x00000010
0x0040127b,	0x00401164,	0x00000010
0x0040116b,	0x00401d30,	0x00000010
0x00401d30,	0x7c810386,	0x00000000
0x7c8103a5,	0x00401170,	0x00000010
0x00401173,	0x00401960,	0x00000010
0x00401996,	0x00401a43,	0x00000000

COSEINC TRACE

libtempltracer.dll

0x004012a0
0x004012b3
0x00401150
0x00401260
0x004013e0
0x004013f4
0x00401417
0x0040140b
0x00401278
0x00401164
0x00401dd0
0x00401170
0x004019e0
0x00401a00
0x00401a0c
0x00401ac3
0x00401a1c
0x00401a28
0x00401a37
0x00401a46
0x00401a55
0x00401a64
0x00401a70
0x00401a86
0x00401a94
0x00401a9f
0x00401ac0
0x00401178
0x00401ae0
0x0040117d
0x00401d68

Post Processing

- What to do with this data?
- Working on visualization techniques for quick ID by analysts(?)
- Malware classification
- Code coverage
- Pretty Pictures

Heatmaps

- Ours are low resolution

MOVIE

DEMO

- Trace Skype
 - See:
 - Needle in the Skype
 - Vanilla Skype

OllyDbg - Skype.exe

File View Debug Plugins Options Window Help

L E M T W H C / K B R ... S

CPU - thread 00000A8C, module ntdll

Address	Disassembly	Registers
7C90EB94	C3 RETN	EAX 00000000
7C90EB95	80A424 00000000 LEA ESP,DWORD PTR SS:[ESP]	ECX 00000000
7C90EB9C	806424 00 LEA ESP,DWORD PTR SS:[ESP]	EDX 7C90E000
7C90EBA0	90 NOP	EBX FFFFFFFF
7C90EBA1	90 NOP	ESP 0576F000
7C90EBA2	90 NOP	EBP 0576F000
7C90EBA3	90 NOP	ESI 00000000
7C90EBA4	90 NOP	EDI 0576F000
7C90EBA5	805424 08 LEA EDX,DWORD PTR SS:[ESP+8]	EIP 7C90E000
7C90EBA9	CD 2E INT 2E	C 0 ES 0
7C90EBAB	C3 RETN	P 1 CS 0
7C90EBAC	55 PUSH EBP	A 0 SS 0
7C90EBAD	8BEC MOV EBP,ESP	Z 0 DS 0
7C90EBAF	9C PUSHFD	S 0 FS 0
7C90EBB0	81EC D0020000 SUB ESP,2D0	T 0 GS 0
7C90EBB6	8985 DCFDFFFF MOV DWORD PTR SS:[EBP-224],EAX	D 0 Last
7C90EBBC	898D DCFDFFFF MOV DWORD PTR SS:[EBP-228],ECX	EFL 00000000
7C90EBC2	8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]	ST0 empty
7C90EBC5	8B4D 04 MOV ECX,DWORD PTR SS:[EBP+4]	ST1 empty
7C90EBC8	8948 0C MOV DWORD PTR DS:[EAX+C],ECX	ST2 empty
7C90EBCB	8085 2CFDFFFF LEA EAX,DWORD PTR SS:[EBP-2D4]	ST3 empty
7C90EBD1	8988 B8000000 MOV DWORD PTR DS:[EAX+B8],ECX	ST4 empty
7C90EBD7	8998 A4000000 MOV DWORD PTR DS:[EAX+A4],EBX	ST5 empty
7C90EBDD	8990 A8000000 MOV DWORD PTR DS:[EAX+A8],EDX	ST6 empty
7C90EBE3	89B0 A0000000 MOV DWORD PTR DS:[EAX+A0],ESI	ST7 empty
7C90EBE9	89B8 9C000000 MOV DWORD PTR DS:[EAX+9C],EDI	FST 0000
7C90EBEF	804D 0C LEA ECX,DWORD PTR SS:[EBP+C]	FCW 027F
7C90EBF2	8988 C4000000 MOV DWORD PTR DS:[EAX+C4],ECX	
7C90EBF8	8B4D 00 MOV ECX,DWORD PTR SS:[EBP]	
7C90EBFB	8988 B4000000 MOV DWORD PTR DS:[EAX+B4],ECX	
7C90EC01	8B4D FC MOV ECX,DWORD PTR SS:[EBP-4]	
7C90EC04	8988 C0000000 MOV DWORD PTR DS:[EAX+C0],ECX	
7C90EC0A	8C88 BC000000 MOV WORD PTR DS:[EAX+BC],CS	
7C90EC10	8C98 98000000 MOV WORD PTR DS:[EAX+98],DS	
7C90EC16	8C80 94000000 MOV WORD PTR DS:[EAX+94],ES	
7C90EC1C	8CA0 90000000 MOV WORD PTR DS:[EAX+90],FS	
7C90EC22	8CA8 8C000000 MOV WORD PTR DS:[EAX+8C],GS	
7C90EC28	8C90 C8000000 MOV WORD PTR DS:[EAX+C8],SS	
7C90EC2E	C700 07000100 MOV DWORD PTR DS:[EAX],10007	
7C90EC34	6A 01 PUSH 1	
7C90EC36	50 PUSH EAX	

Process terminated, exit code C0000005 (-1073741819.)

Terminated

PhantOm

File View Debug Plugins Options Window Help

LEMTWHC / KBR ... S

C:_O - main thr3ad, module ntdll

Memory map

Address	Size	Owner	Section	Contains	Type	Access
00340000	0000A000					
003500						
003600						
003700						
003800						
003900						
003A00						
003B00						
003C00						
003D00						
003E00						
003F00						
004000						
004100						
00F720						

Dump - Skype.text 00401000..00F71FFF

Address	Hex dump	ASCII
00F78000	00 00 00 00 00 00 00 00	
00F78010	9E B1 EC 24 02 00 40 00	Wp500
00F78020	10 51 41 00 1C E2 41 00	0A.LBA
00F78030	24 54 41 00 A0 96 41 00	TA.AA
00F78040	C0 8D 41 00 72 12 8B C0	LA.r#l
00F78050	00 8D 40 00 01 8D 40 00	.ie.0ie
00F78060	00 8D 40 00 01 8D 40 00	.ie.0ie
00F78070	00 00 00 00 00 00 00 00	
00F78080	C0 87 F7 00 50 4E 38 02	++PH;0
00F78090	AC 15 40 00 00 FF 10 00	%S0
00F780A0	D0 4C 38 02 00 4E 77 06	AL;0#Nu#
00F780B0	C0 9B 77 06 A0 28 77 06	Lu#(u#
00F780C0	A0 28 77 06 30 0B 30 73	al(u#00s
00F780D0	0C 16 40 00 00 FF 18 00	0

Registers (FPU)

Register	Value
ERX	A1602938
ECX	AC2C4BA8
EDX	03786665
EBX	02385190
ESP	0012F740
ESI	000003C4
EDI	00000000
EIP	7C90EB94 ntdll.KiFastS
C 1	ES 0023 32bit 0(FFFFFFF
P 1	CS 001B 32bit 0(FFFFFFF
A 1	SS 0023 32bit 0(FFFFFFF
Z 0	DS 0023 32bit 0(FFFFFFF
S 1	FS 003B 32bit 7FFDD000
T 0	GS 0000 NULL
D 0	
O 0	LastErr ERROR_SUCCESS
EFL	00200297 (NO,B,NE,BE,S
ST0	empty 0.3787999999999999
ST1	empty 10.0000000000000000
ST2	empty 0.3787999999999999
ST3	empty 1.0000000000000000
ST4	empty 7.310575179138412
ST5	empty 8.245886882868196
ST6	empty 5.318823021040055
ST7	empty 7.886488245619673
FST	1920 Cond 0 0 0 1 Err
FCW	1272 Prec NERR,53 Mat

Thread 00000F08 terminated, exit code 1E1B (7707).

Running

Test Service

+ Add people

<http://www.skype.com/go/help>

echo123

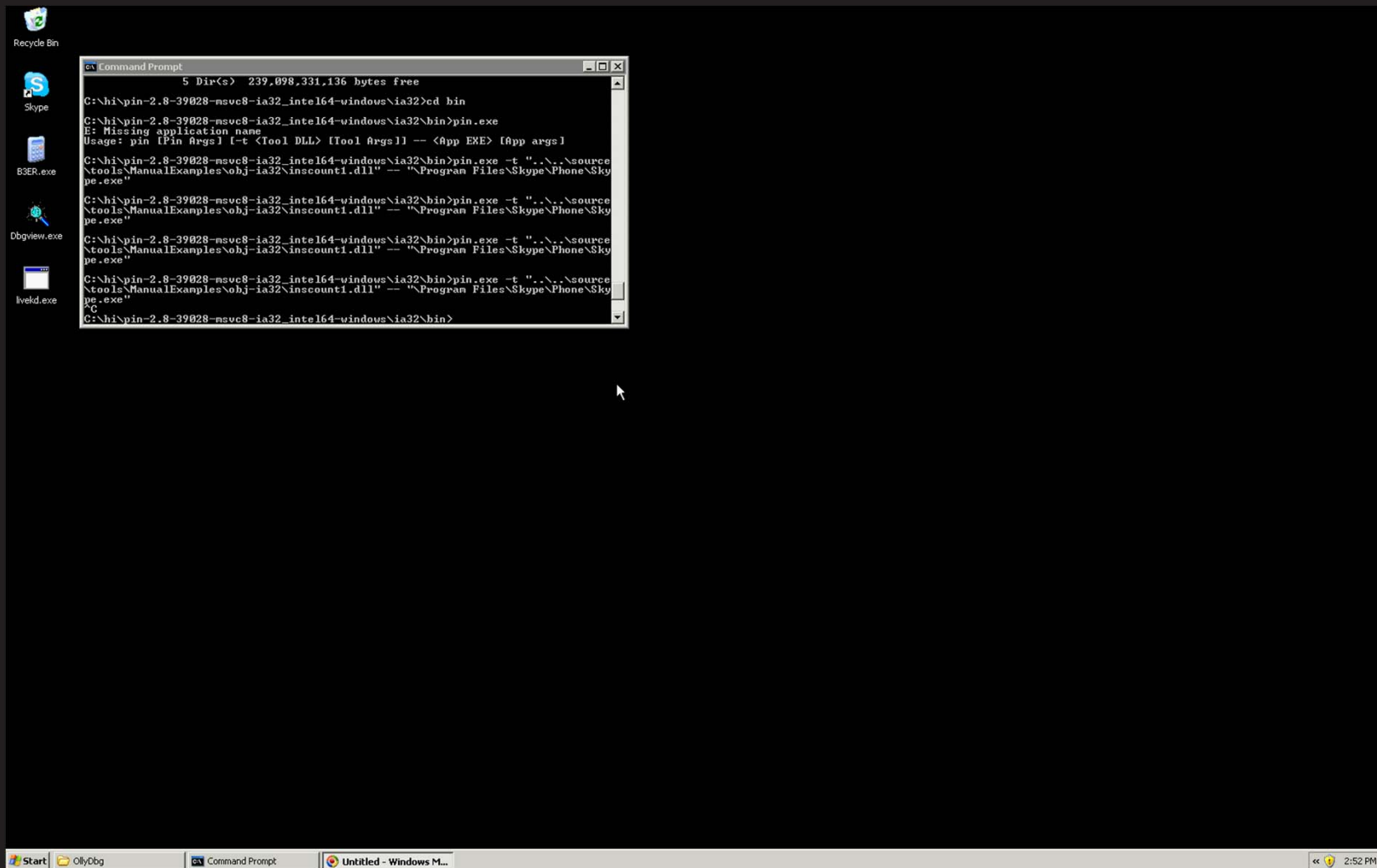
Hi, this is Skype automatic sound test service. Add me to yo...

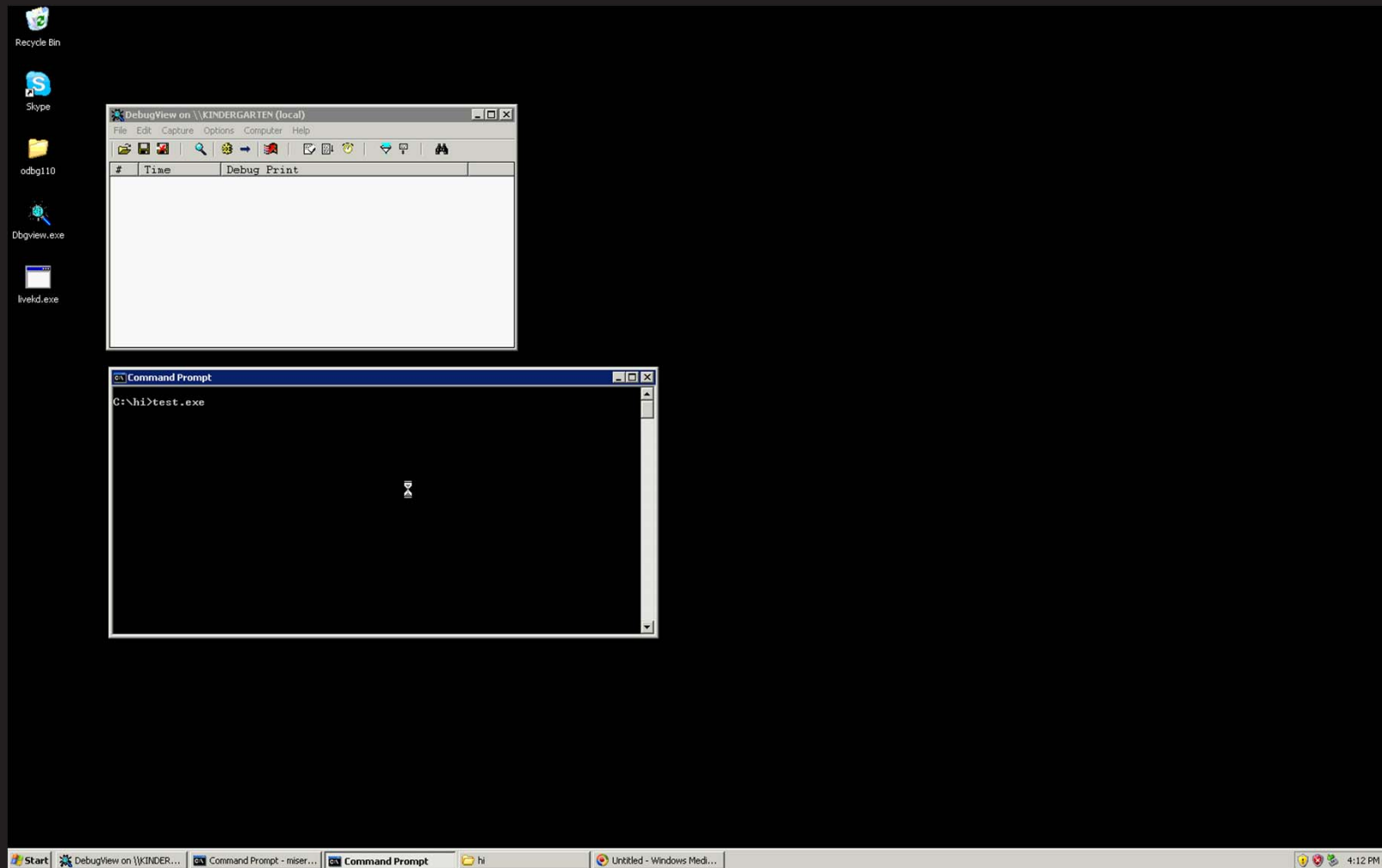
3:12 PM C

3:16 PM C

Extras

o / Sound Test Service here





Aside

- Yet another way to detect if executing in virtual/emulated environment (on netburst):

`wrmsr(MSR_DEBUGCTLA, 1)`

`rdmsr(MSR_LASTBRANCH_TOS)`

Detection

- rdmsr(MSR_DEBUGCTLA)
- rdmsr(IA32_DS_AREA)
- ...

Extensions

- Full fledged kernel debugger leveraging Branch Tracing/PEBS
- Huffman compression of in-kernel branch records
 - Fucking loops
- Using compressibility to classify traces
- IDA scripts (lame)

Related

- ptrace BTS implementation
 - Done by Markus Metzger of Intel
 - Haven't tested it
- Vtune
 - Commercial Intel App.
 - I'm guessing it leverages LBR (don't have \$900)

Related

- Process Stalker
 - Trap On Branch (BTF)
- Saffron
 - ring0 dynamic instrumentation with pin
 - Driver source not released
 - Page faults
- Azure
 - Virtual machine introspection

Off Topic: instrumentation

- Injecting dlls
- Injecting calls
- Is DynamoRio optimizing execution?
 - Are your basic blocks right?
- Exploitability skew?

Greetz

- zen, josh, sam, kat, jan, #social
- Dude at Intel who answered my stupid questions
- Coseinc